**Esercizio 1 - Flywheight pattern - Stampante a colori**

Requisito:

Lo scopo è realizzare il Firmware di una stampante che esegua la stampa di un testo a colori, avvalendosi del pattern Flyweight per ridurre al minimo l’occupazione di memoria di ogni singola sessione di stampa

Il testo da stampare (con i relativi colori) è il seguente

**Hello** **World !**

La simulazione dell’output della stampante sarà affidata alla Console della vostra applicazione.

Suggerimenti:

1. Ogni oggetto può essere modellato da una classe che contenga:
   * Il carattere stesso
   * Il colore 🡪 emumerato ConsoleColor
2. Il Flyweight permette di istanziare un solo oggetto per ogni tipo di combinazione carattere-colore, anche qualora lo stesso appaia più volte all’interno del testo

**Esercizio 2 – Flyweight e abstract Factory**

Requisito:

L’esercitazione punta a creare una sorta di piccolo videogico. Lo scopo dell’applicazione sarà:

* Creare tre personaggi
* Creare delle armi e assegnarle ai personaggi
* Ogni personaggio può portare solo un’arma per tipo. Qualora raccogliesse la stessa arma più volte, le munizioni dovranno essere incrementate di 10 unità
* Ogni personaggio dovrà avere un’arma di default

Di seguito la lista dei personaggi e dei tipi di armi

public enum HeroClass

{

Tank,

Damage,

Support

}

public enum WeaponType

{

Knife,

Gun,

AssaultRifle,

SniperRifle

}

Suggerimenti

1. Il Factory è un pattern “creazionale”, può essere quindi utilizzato per la creazione di personaggi e armi
2. Il Flyweight è pattern “strutturale”, utile quindi a definire la struttura delle calassi e degli oggetti. Si può utilizzare con la

**Esercizio 3 – Strategy**

Riallacciandosi all’esercizio visto in precedenza, agire sulle classi concrete che modellano i player assegnando dei comportamenti

In sostanza ciascun personaggio dovrà implementare i seguenti comportamenti

* Muoversi
* Correre
* Sparare con una delle armi nel suo inventario

La classe “Support” dovrà implementare un’azione specifica

* Cura un altro personaggio

La classe “Damage” dovrà implementare un’azione specifica

* Lancia Granata

Per le azioni specifiche, dovranno essere creati due nuovi oggetti da mettere in inventario. L’enumerato è il seguente

public enum InventoryItem

{

Medikit,

Granade

}

**Esercizio 4 – Singleton**

Uniamo infine quanto visto nelle precedenti esercitazioni, creando un game manager che gestisca il nostro gioco.

Il manager dovrà eseguire le seguenti operazioni.

* Istanziare un livello di gioco soltanto una volta, creando i personaggi e assegnando gli oggetti
* Eseguire una ciclo di 4 turni durante il quale ciascuno dei personaggi eseguirà un’azione tra quelle diponibili